11-1-2005

A new design methodology for mixed level and mixed signal simulation using PSpice A/D and VHDL

Sreeram Rajagopalan

A New Design Methodology for Mixed Level and Mixed Signal Simulation using PSpice A/D and VHDL

by

Sreeram Rajagopalan

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Marcin Lukowiak
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
November 2005

Approved By:

11/15/2005

Dr. Marcin Lukowiak
Visiting Instructional Faculty, Department of Computer Engineering, RIT
Primary Adviser

11/15/2005

Dr. Pratapa Reddy
Professor
Department of Computer Engineering, RIT

11/15/05

Dr. Fei Hu
Assistant Professor
Department of Computer Engineering, RIT

Thesis Release Permission Form

Rochester Institute of Technology

Kate Gleason College of Engineering

Title: A New Design Methodology for Mixed Level and Mixed Signal Simulation using PSpice A/D and VHDL

I, Sreeram Rajagopalan, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Sreeram Rajagopalan

11/16/05

Date

Dedication

To my parents, Gomathy and Rajagopalan, for their love, trust and motivation, without which it would not have been possible.

Acknowledgments

I would like to take this opportunity to thank my advisor Dr. Marcin Lukowiak, for his support, guidance and patience that helped me complete this thesis. I would also like to thank my committee members Dr. Pratapa Reddy, Dr. Fei Hu for their suggestions and Dr. Edward Chueng at NASA Goddard Space Flight Center for trying the tools that I had developed and for letting me share the simulation results in this thesis.

I would like to extend my gratitude to Mr. Manny Marcano, President of EMA Design Automation Inc, along with Mr. Greg Roberts, Director of Marketing and Mr. Bob Kelleher, Director of Sales, for their initiatives and for providing me the resources that were required during this thesis. Miss Reema Divatia and Mr. Nikhil Punnakali, Technical Research Engineers at EMA Design Automation for helping me understand PSpice. I would also like to thank Synplicity Corporation for loaning me a copy of their synthesis tool, Synplify. Finally, I would like to thank all my friends for being there for me.

Abstract

PSpice A/D is a simulation package that is used to analyze and predict the performance of analog and mixed signal circuits. It is very popular especially among Printed Circuit Board (PCB) engineers to verify board level designs. However, PSpice A/D currently lacks the ability to simulate analog components connected to digital circuits that are modeled using Hardware Descriptive Languages (HDLs), such as VHDL and Verilog HDL. Simulation of HDL models in PSpice A/D is necessary to verify mixed signal PCBs where programmable logic devices like Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are connected to discrete analog components. More than 60% of the PCBs that are designed today contain at least one FPGA or CPLD.

This thesis investigates the possibility of simulating VHDL models in PSpice A/D. A new design methodology and the necessary tools to achieve this goal are presented. The new design methodology achieves total system verification at PCB level. Total system verification reduces design failures and hence increases reliability. It also allows reducing the overall time to market. A mixed signal design from NASA Goddard Space Flight Center for a brushless three phase motor that runs a space application is implemented by following the proposed design methodology.

Contents

List of Figures

List of Tables

Glossary

Complex Programmable Logic Device (CPLD) Complex Programmable Logic Device are made up of several simple PLDs (SPLDs) with a programmable switching matrix in between the logic blocks. CPLDs typically use EEPROM, flash memory or SRAM to hold the logic design interconnections.

Electronic Design Automation (EDA) Electronic design automation is the category of tools for designing and producing electronic systems ranging from printed circuit boards (PCBs) to integrated circuits. This is sometimes referred to as ECAD (Electronic Computer-Aided Design). The term EDA is also used as an umbrella term for computer-aided engineering, computer-aided design and computer-aided manufacturing of electronics in the discipline of electrical engineering.

Field-Programmable Gate Array (FPGA) An FPGA is a semiconductor device used to process digital information, similar to a microprocessor. It utilizes gate array technology that can be reprogrammed after it is manufactured, rather than having its programming fixed during the manufacturing. FPGAs belong to the family of Programmable Logic Devices (PLD)

Hardware Descriptive Language (HDL) Hardware Descriptive Language is any language from a class of computer languages used for formal description of electronic circuits. It can describe the circuit's operation, its design, and tests to verify its operation by means of simulation.

Johnson Counter A Johnson counter is a special case of shift register, where the output from the last stage is inverted and fed back as input to the first stage. An n-stage Johnson counter yields a count sequence of length 2n, so it may be considered to be a mod-2n counter.

MATLAB SimuLink MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran. Simulink is a platform within Matlab commonly used for electro-mechanical and other system level design and simulation.

ModelSim ModelSim is a UNIX, Linux, and Windows-based, VHDL and Verilog simulation environment. It is currently owned by Mentor Graphics Corporation, USA.

National Aeronautics and Space Administration (NASA) The National Aeronautics and Space Administration (NASA) (established 1958) is the government agency responsible for the United States of America's space program and long-term general aerospace research.

A list of logic gates and their interconnections which make up a circuit. It intends to describe the connectivity of an electronic design.

Printed Circuit Board (PCB) Printed Circuit Board is a flat, insulated base board on which the metallic conducting paths connecting circuit components are fixed. The base is typically of plastic, glass, ceramic, or some other dielectric, and the conducting paths may be placed on it by a variety of methods. The circuit components like resistors, capacitors, and other devices are mounted on the finished base board either with their leads being inserted through holes drilled through both, the conducting pattern and the base and soldered to the conducting strips or, increasingly, by directly soldering leadless components to the circuit.

Programmable Logic Device (PLD) Programmable Logic Devices (PLDs) are a generic term for an integrated circuit that can be programmed or configured to perform complex functions.

PSpice A/D PSpice A/D is a simulation program that models the behavior of analog, digital or mixed analog/digital circuits. PSpice was commercially introduced in 1985 and it is currently a part of Cadence Design systems

PSpice Subcircuit A subcircuit definition contains Spice circuit elements, has a name and specifies the circuit nodes that connect it to the main circuit.

Register Transfer Level (RTL) Register transfer level description is a description of a digital electronic circuit in terms of data flow between registers, which store information between clock cycles in a digital circuit. Logic synthesis tools may be used to automatically convert the RTL description of a digital system into a gate level description of the system.

Synplify A high-performance logic synthesis engine that delivers fast, highly efficient FPGA and CPLD designs. Synplify takes Verilog and VHDL Hardware Description Languages as input and outputs an optimized netlist in most popular FPGA vendor formats. It is currently owned by Synplicity Corporation, USA.

Logic Synthesis Logic synthesis is an automated process of generating gate level netlist from a RTL description written in HDLs. Every logic synthesis program understands some subset of Verilog and VHDL.

VHDL VHDL stands for Very High Speed Integrated Circuit (VHSIC) Hardware Description Language. It was An IEEE-standard hardware description language originally developed by the US Department of Defense as a common means of documenting electronic systems. Specified in the IEEE 1076 standard and used by electronic designers to describe and simulate their chips and systems prior to fabrication, an alternative language to Verilog.

Chapter 1

Introduction

Printed Circuit Board (PCB) is a flat, insulated base board on which the metallic conducting paths connecting circuit components are affixed. Over the past few decades, PCBs have brought significant change and advancement to the electronics industry. Today, the computers we use, calculators, personal entertainment devices, medical equipments, *etc.*all contain PCBs. Their overall cost, shape and size is dependent on the size and complexity of the PCB that is utilized. A complex board can contain printed circuitry on both its sides or in many layers, which allows great circuit density and compactness. With the increase in complexity and reduced time to market, Computer Aided Design (CAD) tools are being used to design and simulate PCBs. These tools are not only helpful in reducing the time to market a design but also in making them more accurate and stable than those that are hand made [7]. A collection of CAD tools that enhance and aid in development of complex electronic systems are called Electronic Design Automation (EDA) tools.

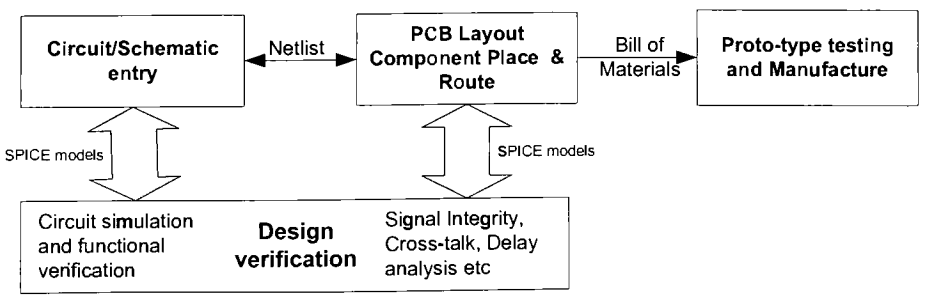


Figure 1.1: Typical PCB Design flow

An overview of a typical PCB design process is presented in figure 1.1. Various EDA tools assist designers during different stages of the design flow. The process begins with describing the design using a schematic/circuit editor, which establishes the connections between different parts of the design. Typically, the next step is to verify the design's functionality by simulating it with appropriate input values. This is often called the front end of the design process and the EDA tools used during this process are called front end tools. Once the functionality of the design is verified, it is now ready to be laid out on a PCB. A list of parts and their connecting points are sent to the PCB layout tool which defines the exact location of design components as it would be on a physical PCB. The usual goal is to reduce the overall area of the PCB. Depending upon the application for which the PCB was designed it is further subjected to post layout analysis to determine phenomenons such as cross talks, low signal strengths, delays, impedance mismatches etc. The process of defining the physical layout of the PCB beginning with the layout tool is called the back end process and the EDA tools used are called back end tools.

Circuit simulation is one of the most time intensive tasks during the PCB design flow. They allow designers to verify whether the circuits function as designed. Simulation results also predicts the performance of the design before committing to large scale manufacturing of PCBs. Complexity of the design, necessity to increase design efficiency and to reduce time to design, drives the circuit design methodology and the choice of EDA tool used during circuit simulation. Design methodologies are typically classified into

Top down design methodology enables designers to refine an abstract idea progressively as the design process continues. Typically, beginning with a definition of the system in a very high level behaviorial notation and then getting down to finer details with Register Transfer Level(RTL) and gate level descriptions, as the design progresses. This methodology is more popular with digital circuit designs with the advent of HDLs, Programmable Logic Devices (PLDs) and logic synthesis tools [9].

Traditional or bottom up design methodology allows designers to pick their

components individually and build the design by connecting them appropriately. This methodology is popular in the PCB design flow [11].

In order to decrease development time, large and complex designs are usually broken into smaller units. These units can be designed using different methodologies. Due to difference in levels of design abstraction, different EDA tools are required to work with different design methodologies. With different EDA tools being used for designing different sections of a system, it is difficult to perform a complete verification of the entire system. Inability to verify the functionality of a design is one of the potential causes for failure. Moreover, a majority of today's designs are mixed signal circuits, circuits with different signal domains *(eg., analog and digital)*. A typical example of such a design would be a PCB which has PLDs along with discrete analog components. In awake of such scenarios, rises a need for EDA tools that are capable of simulating mixed signal designs as well as designs designed using different methodologies. Such simulations are called mixed level and mixed signal simulation.

PSpice A/D has been established as an industry standard in mixed signal simulation with more than 70,000 users in United States alone [10]. It currently supports mixed signal simulation using traditional design methodology, however lacks the ability to simulate digital designs modeled using HDLs such as VHDL, Verilog etc.

Designing with VHDL validates the true value of top down design methodology for digital designs. It provides expressive facilities that helps designers to create state-of-the-art designs.

The objective of this thesis is to enable efficient circuit simulation and thus improve the overall PCB design process. By enabling simulation of VHDL models in PSpice A/D it is possible to realize a mixed level simulator from a mixed signal simulator. This integrates traditional designing methodology with top down design methodology, which helps in verifying the functionality of the whole system and identifying problems much earlier in the design cycles.

Chapter 2

Background

2.1 Circuit simulation using PSpice A/D

SPICE stands for Simulation Program with Integrated Circuit Emphasis. It is a powerful software tool used to model and analyze electric circuits. It was developed in the early 1970s in the Department of Electrical Engineering at University of California at Berkeley, progressing through various versions and culminating in 1981 with SPICE 2G.6, which is the version on which most commercial simulation packages are based. Although SPICE 2G.6 is a powerful and robust program, it was not commercially successful for a number of reasons. The early versions needed mainframe computers and had unfriendly user interfaces. They were not interactive and lacked the ability to simulate mixed signal circuits, finally there were no component libraries and all these features made the original SPICE unattractive. A number of commercial organizations noticed the potential of the core software, provided that these short comings were rectified.

With its inception in 1985, PSpice A/D (now a part of Cadence design systems) is an industry standard for circuit simulation packages. Its latest version provides an interactive graphic user interface for the traditional design methodology using SPICE2G.6, where designer connects components by picking them from a given library. PSpice A/D is used during the functional verification of a circuit design before committing to PCB layout. Figure 2.1 presents an analog circuit whose input and output characteristics are analyzed using PSpice simulation results.

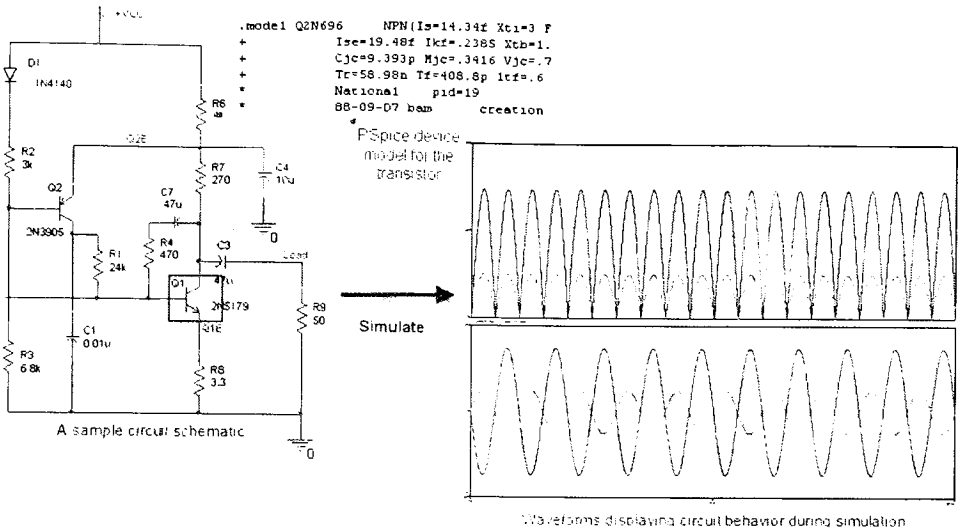


Figure 2.1: PSpice simulation of a circuit

The standard PSpice A/D libraries feature over 16000 analog and 1600 digital devices, manufactured in North America, Japan and Europe. A device library is in fact, a collection of *model* files. The model file defines electrical characteristic of a part, which is be used during simulation to determine its response to different electrical or logical inputs. All components shown in circuit of figure 2.1 have an associated PSpice device model.

2.1.1 Modeling of digital devices in PSpice A/D

Each digital part is described electrically by a digital device model which is stored within a model library. The standard PSpice library of digital devices contains parts from different technologies (e.g.*TTL, CMOS, ECL*). A complete digital device model has three main characteristics:

- Functional behavior: described by gate level and behavioral digital primitives comprising the subcircuit
- The timing behavior: described by specifying propagation delays and timing constraints such as setup and hold times.

- The I/O behavior: described by information specific to the device's input/output characteristics.

The functional behavior of a digital device is defined by one or more interconnected digital primitives. Typically, logic diagrams of digital circuits can be implemented by interconnecting digital primitives that are available with PSpice A/D. Table 2.1 shows some of the available digital primitives in PSpice A/D [5].

Type	Definition
BUF	Buffer
INV	Inverter
AND	AND gate
OR	OR gate
NOR	NOR gate
NAND	NAND gate
XOR	Exclusive OR gate
NXOR	Exclusive NOR gate
BUFA	Buffer array
ANDA	AND array
XORA	Exclusive OR gate array
AO	AND-OR Compound gate
OA	OR-AND Compound gate
AOI	AND-NOR Compound gate
JKFF	J-K, negative edge triggered flip flop
DFF	D-type, positive edge triggered flip flop
SRFF	S-R gated latch
DLTCG	D gated latch

Table 2.1: Examples of currently available PSpice digital primitives

The timing behavior describes a digital component's timing characteristics, such as

- Propagation delays (TP)
- Setup times (TSU)
- Hold times (TH)
- Pulse widths (TW)

- Switching times (TSW)

Each of these parameters are further divided into three values

- Minimum (MN)
- Typical (TY)
- Maximum (MX).

For example, the typical low-to-high propagation delay on a gate is specified as TPLHTY. The minimum data-to-clock setup time on a flip-flop is specified as TSUDCLKMN. Figure 2.2 shows sample values for different timing parameters for an edge triggered flip-flop within a device timing model.

```
.model D_393_1 ueff (
+  tppcqhlty=18ns    tppcqhlmx=33ns
+  tpclkqlhty=6ns    tpclkqlhmx=14ns
+  tpclkqhlty=7ns    tpclkqhlmx=14ns
+  twclkhmn=20ns     twclklmn=20ns
+  twpclmn=20ns      tsudclkmn=25ns
+  )
```

Figure 2.2: A sample timing model for an edge triggered flip-flop

The Input/Output behavior describes the component's output drive resistance and loading capacitances. Figure 2.3 is an example for input/output model. While the timing infor-

```
.model IO_STD uio (
+  drvh=96.4   drvl=104
+  AtoD1="AtoD_STD"  AtoD2="AtoD_STD_NX"
+  AtoD3="AtoD_STD"  AtoD4="AtoD_STD_NX"
+  DtoA1="DtoA_STD"                DtoA2="DtoA_STD"
+  DtoA3="DtoA_STD"  DtoA4="DtoA_STD"
+  tswhl1=1.373ns          tswlh1=3.382ns
+  tswhl2=1.346ns          tswlh2=3.424ns
+  tswhl3=1.511ns          tswlh3=3.517ns
+  tswhl4=1.487ns          tswlh4=3.564ns
+  )
```

Figure 2.3: A sample Input/Output model

mation is specific to a device, the input/output characteristics are specific to a whole logic

family. Thus, many devices in the same family reference the same I/O model, but each device has its own timing model. It also contains the names of up to four AtoD and DtoA subcircuits that the simulator calls to handle analog/digital interface nodes. Table 2.2 shows a list of parameters that are typically configured by a device input/output model. The pro-

IO model parameter	Description
INLD	Input load capacitance
OUTLD	Output load capacitance
DRVH	Output high level resistance
DRVL	Output low level resistance
INR	Input leakage resistance
DIGPOWER	Digital power supply for A/D interface subcircuits
TSWLH and TSWHL	Switching time (low - high/high -low)
AtoD and DtoA	Name of the interface subcircuits

Table 2.2: Digital I/O model parameters

posed design methodology utilizes a PSpice library of customized digital parts which was developed as a part of this research. These digital parts are modeled by using digital primitives that are currently supported in PSpice A/D (Table 2.1. However, they are modeled with ideal timing and input/output behavior. Hence they exhibit no delays and provide optimum fan-out capabilities.

Digital devices in PSpice are defined using the *subcircuit syntax*. A subcircuit syntax would include:

- Netlists to describe the structure and function of the part.
- Variable input parameters to fine-tune the model.

Figure 2.4 shows an example of a digital device definition with the *subcircuit* format. This example elucidates the construction of a 7400 device, which basically performs a 2 input NAND operation. The inputs are A, B and output is Y. Additional lines in a PSpice subcircuit definition can be added using "+" character in the beginning of a new statement.

The PSpice model description for the NAND gate in figure 2.4 includes an optional device parameter called MNTYMXDLY. This parameter selects either the minimum, typical or

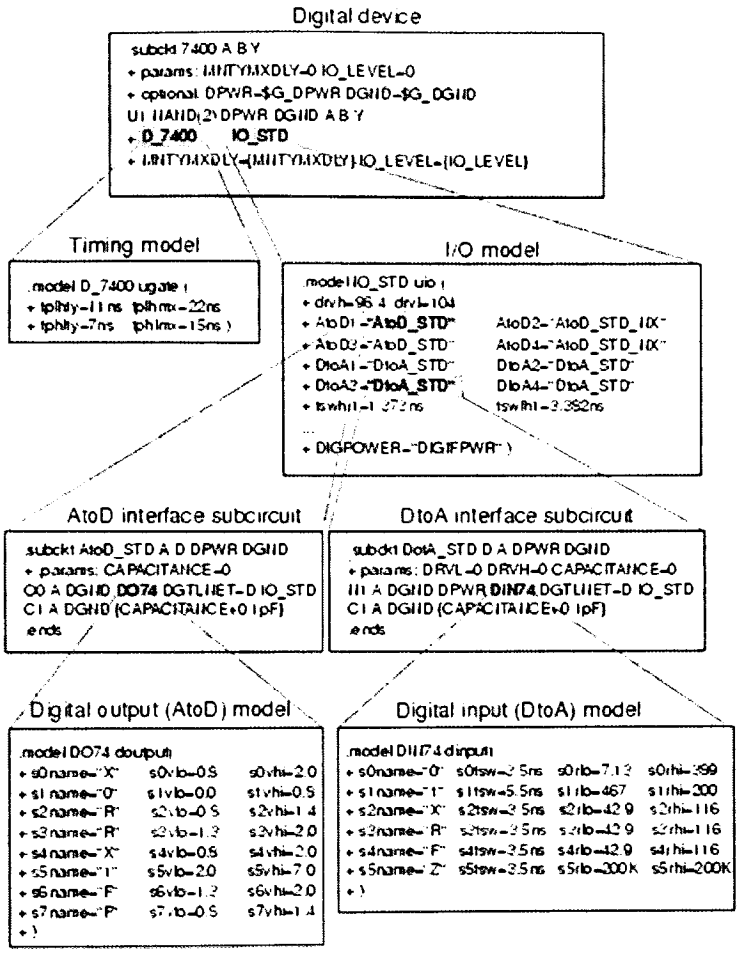


Figure 2.4: Elements of digital device definition

maximum delay values from device timing model. If this is not specified, MNTYMXDLY defaults to 0. Valid values are shown in the table 2.3

IO_LEVEL is an optional device parameter that selects one of the four AtoD or DtoA interface subcircuits from the device's I/O model. PSpice A/D calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are shown in table 2.4

Digital power nodes [$G_DPWR $G_DGND] are the nodes used by the interface subcircuits which connect analog nodes to digital nodes or vice versa.

0	The current value of the circuit-wide DIGMNTYMX option (default=2)
1	Minimum
2	Typical
3	Maximum
4	Worst-case timing

Table 2.3: Valid values for MNTYMXDLY

0	The current value of the circuit-wide DIGIOLVL option (default=1)
1	AtoD1/DtoA1
2	AtoD2/DtoA2
3	AtoD3/DtoA3
4	AtoD4/DtoA4

Table 2.4: Valid values for IO_LEVEL

Timing model, (here D_7400) describes the main subcircuit's timing characteristics, such as propagation delay, set up and hold times etc. Each timing parameter has a minimum, typical and maximum value which may be selected during analysis setup.
I/O model name, (here IO_STD) is the name of an I/O model that describes the devices loading and driving characteristics. I/O models also contains names of up to four DtoA and AtoD interface subcircuits, which are automatically called by PSpice A/D to handle analog/digital interface nodes.

During simulation, the part instances and nets defined in a circuit schematic is translated into parts connected by nodes. The standard simulation netlist contains a flat view of the circuit. PSpice A/D extracts definitions for all the parts modeled as subcircuits, viewing parts as a collection of primitive parts and node connections. The digital primitives that make up a digital part determine the way that PSpice A/D processes an analog/digital interface to that part. Specifically, the I/O model for each digital primitive connected at the interface gives PSpice A/D the necessary information. [4]

2.1.2 Digital simulation in PSpice A/D

Digital simulation is the analysis of logic and timing behavior of digital devices over time. PSpice A/D simulates this behavior during transient analysis. It also performs detailed timing analysis depending upon the constraints specified for the devices.

When the circuit is in operation, digital nodes takes on values or output states as shown in table 2.5. Each digital state has a *strength* component as well. PSpice A/D refers to logic levels and not actual voltages. For example, logic level "1" means only that the voltage is somewhere within the "high" range of a particular device family. The rising and falling edge only indicate that the voltage crosses the 0-1 threshold at some time during the 'R' or 'F' interval, and not that the voltage change follow a particular slope.

State	Meaning
0	Low, false, no, off
1	High, true, yes, on
R	Rising (change from 0 to 1 during rising edge)
F	Falling (change from 1 to 0 during falling edge)
X	Unknown: may be high, low, intermediate or unstable
Z	High impedance: may be high, low, intermediate or unstable

Table 2.5: PSpice digital states

When a digital node is driven by more than one device, PSpice A/D determines the correct level of the node. Each output has a *strength* value, and PSpice A/D compares the strengths of outputs that are driving the node. The *strongest* driver determines the resulting level of the node. If outputs of the same strength but different levels drive a node, the node's level become X.

PSpice A/D supports 64 strengths. The lowest (weakest) strength is called Z. The highest (strongest) strength is called the forcing strength. The 64 strengths are determined by two user specified parameters during simulation (DIGDRVZ and DIGDRVF), which defines the impedance of Z strength, and forcing strength respectively. PSpice A/D then formulates a logarithmic scale consisting of *64 range* of impedance values. The simulator then uses the

values of DRVH (high level driving resistance) or DRVL (low level driving resistance) parameters from the device's I/O model. Drive impedances which are higher than the value of DIGDRVZ are assigned the Z strength (0). Similarly, drive impedances lower than the value of DIGDRVF are assigned the forcing strength (64).

A digital stimulus defines input to digital devices, playing a role similar to independent voltage sources for analog circuits. PSpice provides a library of digital stimuli along with a graphical user interface called the Stimulus Editor that allows the user to define the digital input (Adding/Moving/Editing/Deleting state transitions). However, unlike other digital simulators, PSpice A/D does not allow changing the input levels during simulation runtime. When input values are changed, the PSpice digital simulator will restart the simulation from the beginning (when time = '0').

2.1.3 Mixed signal simulation in PSpice A/D

During mixed signal simulation, PSpice A/D translates all part instances and nets defined in a design into parts connected by nodes. This process is called *netlisting*. It also extracts the definitions of all parts modeled as subcircuits and views them as a collection of primitive parts in their respective nodes. PSpice A/D recognizes three types of nodes: analog nodes, digital nodes and interface nodes. The node type is determined by the type of component connected to it. If all parts connected to a node are analog, then it is an analog node. If all parts are digital, then it is a digital node. If there is a combination of analog and digital parts then it is an interface node. Figure 2.5 is a simple mixed signal circuit. Resistors R1, R2 and capacitor C1 are analog components whereas, U1 represents a digital device.

- Node 1 is an analog node
- Node 2 connects resistor R1, capacitor C1 and a digital device U1, hence this is an interface node
- Node 3 connects a digital device U1 and resistor R2, hence this is an interface node

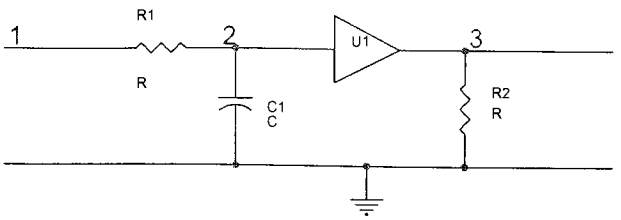


Figure 2.5: A simple mixed signal circuit

Interface nodes are automatically broken into one purely analog or digital node by inserting one or more analog/digital interface subcircuits. These interface subcircuits are called AtoD and DtoA subcircuits. The function of an AtoD interface subcircuit is to change voltages and impedances to digital states as per table 2.5 and similarly, the function of a DtoA subcircuit is to change digital states to voltages and impedances. Figure 2.6 shows the circuit of figure 2.5 after PSpice has inserted AtoD and DtoA interface subcircuits. After the interface generation, the circuit has two new digital nodes designated as 2$AtoD and 3$DtoA, which ensures that all nodes in the circuit receive either pure analog or digital value. PSpice also automatically connects a power supply to the interface subcircuits to complete the generation of the interface.

The standard model library is shipped with interface subcircuits for the following logic families

- TTL
- CD4000 series CMOS and high-speed CMOS (HC/HCT)
- ECL 10k
- ECL 100k

In this research, the interface subcircuits from the TTL family is being utilized with the custom models that are defined. Each digital primitive comprising the subcircuit description of the digital part has an I/O model describing its loading and driving characteristics. The name of the interface subcircuit actually inserted by the PSpice A/D is specified by the

I/O model of the digital primitive at the interface. The I/O model has parameters for up to four analog-to-digital and four digital-to-analog subcircuit names. The parameters are chosen based on the accuracy desired. In this research, level 1 parameters (default) produced required accuracy. For more details on the parameters and available I/O levels, please refer [5].

Digital power supplies are used to power the interface subcircuits that are automatically inserted by PSpice A/D during a mixed signal simulation. It creates the appropriate interface subcircuit and power supply according to the I/O model referenced by the digital part. The I/O model is specific to the digital part's logic family. The power supply provides the necessary reference or drive voltage for the analog side of the interface.

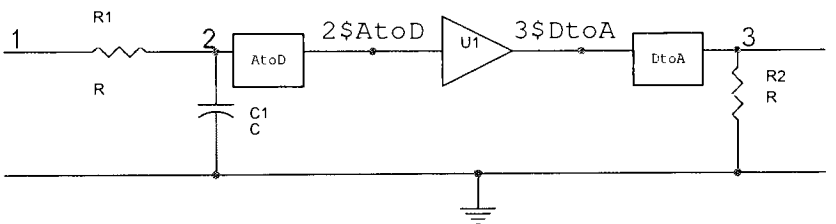


Figure 2.6: Circuit of figure 2.5 after PSpice has inserted AtoD and DtoA interface subcircuits

2.2 VHDL and Logic synthesis

VHDL is a language for describing digital electronic systems. It arose out of the United States government's Very High Speed Integrated Circuits (VHSIC) program. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of Integrated Circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed. It was subsequently developed further under the auspices of the Institute of Electrical and Electronic Engineers (IEEE) and adopted in the form of the IEEE Standard 1076, Standard VHDL Language Reference Manual, in 1987.

2.2.1 Basic structure of a VHDL file

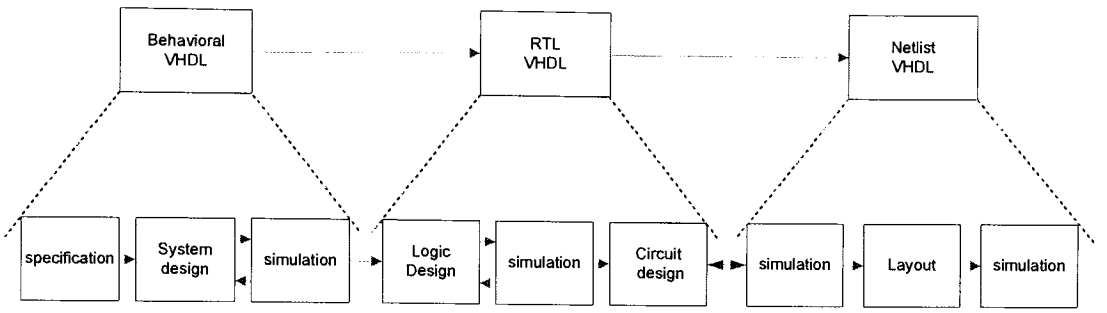


Figure 2.7: Design flow in VHDL

The various design abstractions in VHDL during a typical top down design flow is presented in figure 2.7. Specification of the digital design is defined with respect to function, size, interfaces, etc using high level *behavioral* abstraction. It is simulated to validate functionality. At the Register transfer level (RTL), the models that have to be designed are described with all the synthesis aspects in view. As long as only a certain subset of VHDL constructs is used, commercial synthesis tools can derive the boolean functions from this level and map them to appropriate (logic) elements of a target technology. The result is a *netlist* of the circuit or of the module on the gate level. Every transition to a lower abstraction level is proven by simulation [1]. Software tools such as ModelSIM, NC VHDL etc are available to simulate VHDL.

A digital system in VHDL is expressed with design entities. Entity is the most basic building block in a design, just like a symbol in a traditional design methodology, see figure 2.8. Each entity is modeled by an *entity declaration* and an *architecture body*. The entity declaration contains inputs and output signals and acts like an interface to the outside world, while the architecture body contains the description of the entity and is composed of interconnected entities, processes and components, all operating concurrently. In a typical design there will be many such entities connected together to perform the desired function. Figure 2.9 shows the entity declaration for a four bit parity generator.

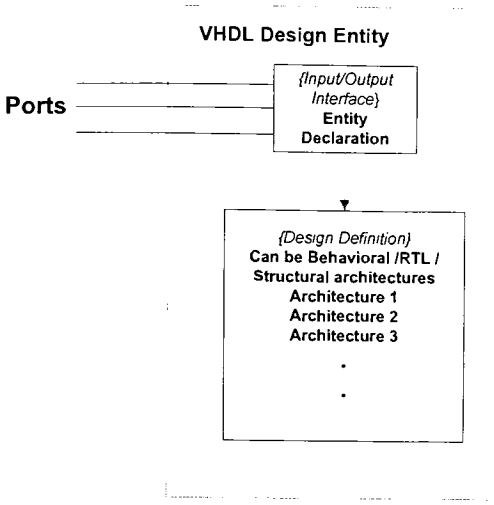


Figure 2.8: A representation of design entity in VHDL


```
entity Parity is
    Port (V: in std_logic_Vector(3 downto 0);
          Even: out std_logic);
end Parity;
```

Figure 2.9: *entity* declaration in VHDL for a 4 bit parity generator

The architecture body specifies how the design operates or how it is implemented. It supports all the three levels of design abstraction: behavioral, RTL, structural, or a combination of all the three. At the Behavioral level, complex algorithms are described without considering hardware implementation. Language constructs like *wait,delay, loops* are used. It is the fastest method to implement and validate the concept of an entire system. Figure 2.10 presents a behavioral modeling in VHDL for the 4 bit parity generator.

At RTL level, the system is described using a subset of VHDL that is suitable for synthesis. It is described in terms of registers and logic that calculates the next value of the storage elements. It is also used to define state machines for controller designs, where state registers may be either explicitly or implicitly defined depending on the coding style. In the above example, the RTL description for the parity generator would be similar to its

```
architecture Parity_Behavioral of Parity is
begin
    Process(V)
        Variable NR_1: Natural;
        begin
            NR_1:=0;
            For I in 3 downto 0 loop
                if V(I)   '1' then
                    NR_1:= NR_1 +1;
                end if;
            end loop;
                If NR_1 mod 2 := 0 then
                    Even <= '1' after 2.5 ns;
                else
                    Even <= '0' after 2.5 ns;
                end if;
    end process;
end Parity_Behavioral;
```

Figure 2.10: Behavioral VHDL description of the 4 bit parity generator

behavioral description, but without any delays.

Figure 2.11 represents the 4 bit parity generator in terms of basic gates. This level of VHDL abstraction is called structural description, see figure 2.12. Although it is possible to define designs at this level, in practice a gate level netlist (structural VHDL) is generated from the RTL description, with the help of synthesis tools. For this task, a cell library for the target technology which holds the information about all available gates and their parameters is required.

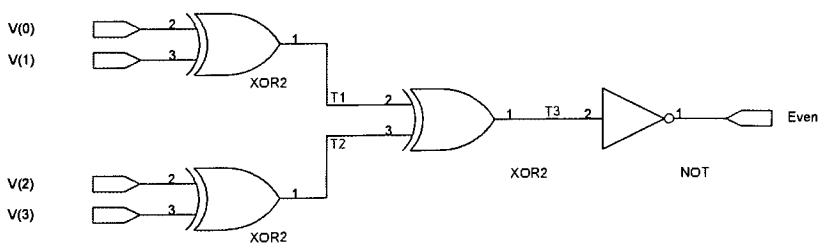


Figure 2.11: Parity generator represented by logic gates

```
use WORK.all;
architecture Parity_Structural of Parity is
    component XOR_Gate
        port ( X,Y : in std_logic;
                 Z : out std_logic);
    end component;

    component INV
    generic (DEL: Time);
        port ( X : in std_logic;
               Z : out std_logic);
    end component;

signal T1, T2, T3 : std_logic;

begin
    XOR1: XOR_Gate port map (V(0), V(1), T1);
    XOR2: XOR_Gate port map (V(2), V(3), T2);
    XOR2: XOR_Gate port map (T1, T2, T3);
    INV1: INV
        generic map (0.5 ns)
        port map (T3, Even);
end Parity_Structural;
```

Figure 2.12: Structural VHDL description of the 4 bit parity generator

2.2.2 Logic synthesis

The real keystone for top down design methodologies using HDLs is logic synthesis [8]. This is an automated process of generating gate level netlist (for implementation on a targeted technology) from a RTL description written in HDLs . Typically, logic synthesis has two processes: Level translation and Logic optimization. Level translation refers to the mapping of RTL circuit description into a gate level netlist for the targeted technology. During the optimization phase, the tool searches for equivalent circuits that would reduce the area or increase the performance of the actual implementation. If the target architecture is a Complex Programmable Logic Device (CPLD) with AND-OR plane logic, the optimization step would involve finding the simplest sum-of-product representation for the circuit's logic function. Once synthesis is completed, architecture specific tools, such as placement and routing tools, are required to complete the implementation of the circuit. Figure 2.13 is an overview of earlier hardware design flow in which synthesis is the crux. There are several synthesis tools available from different design automation tool vendors. While many of them perform the same general process, they differ in their optimization

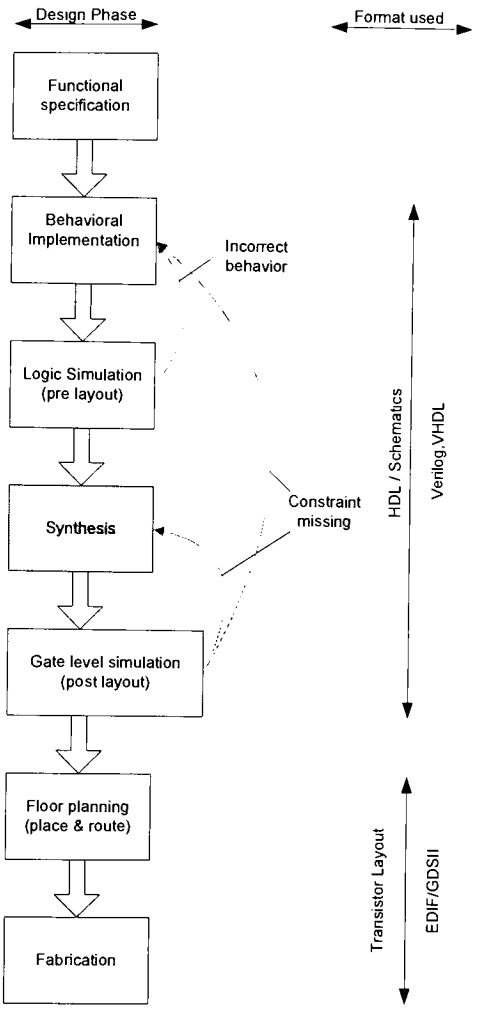


Figure 2.13: Synthesis flow diagram

method and the way in which it allows users to specify synthesis constraints.

2.3 Programmable Logic Devices in PCB design flow

Programmable Logic Devices (PLDs) are a generic term for an integrated circuit that can be programmed or configured to perform complex functions. PLDs can be classified into

1. Simple Programmable Logic Devices (SPLDs)

Programmable Logic Arrays (PLAs), Programmable Array Logic (PALs), Programmable Logic Only Memory (PROM), Generic Array Logic (GAL)

2. Complex Programmable Logic Devices (CPLDs)

3. Field Programmable Gate Arrays (FPGAs)

Amongst the various PLDs that are available in the market today, FPGAs are the most complex. They offer high density along with good flexibility and speed. Since their introduction in 1984, it has become one of the most popular implementation media for digital circuits and have grown into a $ 2 billion per year industry since the early ninties [2]. Modern FPGAs usually have additional embedded resources like configurable memories, specialized multipliers, fast carry logic etc. HDLs and logic synthesis (top down design methodology) are predominantly used in their design process. Today, more than 60 percent of the PCBs include atleast one CPLD or FPGA in their bill of materials. [3]

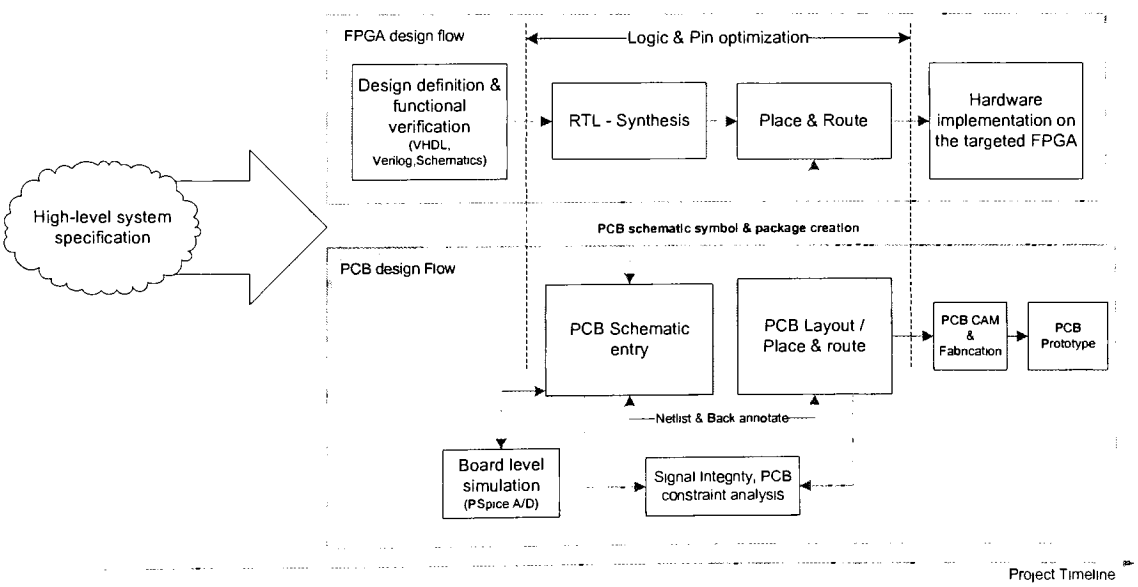


Figure 2.14: FPGAs in PCB Design flow

Figure 2.14 presents a design flow for PCBs that contain FPGAs. In most design environments, the designing of PCBs and FPGAs occur simultaneously and independently.

Multi-million gate FPGAs are typically designed by teams. Teams also design complex PCBs. In many corporations these teams are independent arms of the engineering organization and are often geographically dispersed. It is common to see poor communication between these teams because the design methodology for an FPGA is different from that of a PCB [3].

The electrical properties of an FPGA is integrated into the PCB design flow by the schematic entry tools. A schematic symbol representing the functionality of the FPGA is necessary to establish it's connectivity with other discrete components present on the PCB. For creating a schematic symbol, the PCB designers require input and output properties of the FPGA. Therefore, the FPGA designers would have to complete logic synthesis and, place and route operations before the schematic is ready for PCB layout.

An FPGA designer can completely reconfigure the FPGA and change it's input/output characteristics. A simple change in an FPGA's pin assignment may require drastic changes in the PCB physical design, and in many cases requires the PCB designer to start over. During the circuit simulation stage, most EDA tools lack the ability to verify the functionality of the FPGA along with other analog or digital components. This is primarily because a PCB is designed using traditional design methodology, where as the functionality within an FPGA is implemented by following top down design methodology. After circuit simulation, the back end process would then follow the typical PCB design flow. In general, changes to FPGAs must be communicated to the PCB design flow, be evaluated within the PCB design context, and then any necessary changes must be communicated back to the FPGA design process, to achieve efficiency in a team based design environment.

Chapter 3

Simulation of VHDL models in PSpice A/D

3.1 Outline

Sections 2.1 and 2.2 of chapter 2 presented the necessary background to understand the difference between PSpice, and hardware descriptive language VHDL. This chapter presents a new design methodology that enables mixed signal simulation in PSpice A/D using VHDL (for digital logic which is to be implemented on PLDs) along with discrete analog components. This process, as explained earlier in Chapter 1, is called mixed level simulation in a mixed signal simulator.

Figure 3.1 has two branches, namely analog and digital. While the analog circuits are designed by following the traditional design methodology in PSpice A/D schematic editor, the digital portion in VHDL follows the top down design methodology. Finally, the interfacing software abridges the two design methodologies by enabling functional verification of the mixed signal design in PSpice A/D.

PSpice A/D does not offer as many levels of design abstractions as offered by VHDL and most of its digital constructs are available only at gate level. Therefore to conduct digital simulation in PSpice A/D, the VHDL model has to be at the gate level. However, in order to retain the essence and rapidness of top down design methodology in digital designs, and to ease the digital designer with the ability to describe high level abstract models in VHDL and be less concerned about actual details, the proposed flow includes a Synthesis (Synplify) tool which would take care of the gate level details.

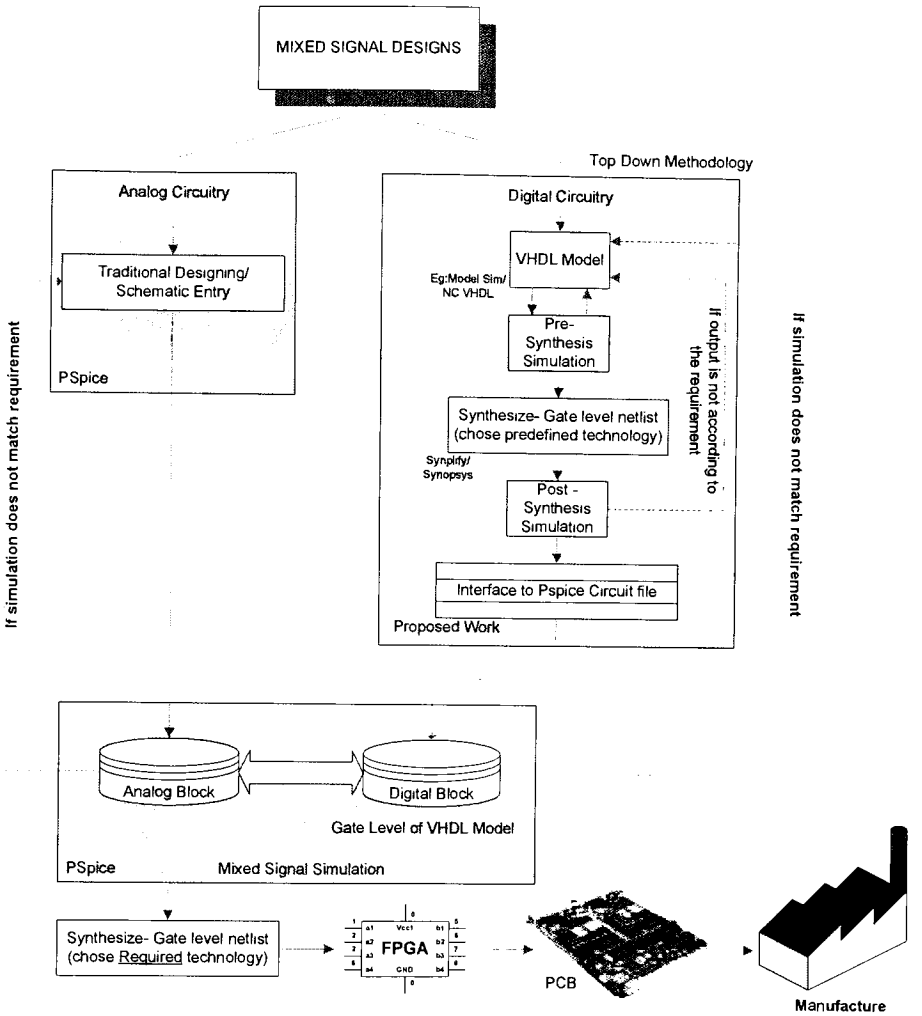


Figure 3.1: Proposed design methodology

A synthesized VHDL code (gate level netlist) represents the digital system in terms of logic gates. In order to simulate this gate level VHDL netlist in PSpice A/D, it needs to be in a format that is understood by the PSpice simulation engine. In other words, the gate level VHDL netlist requires to be translated into a SPICE circuit file or netlist.

The gate level VHDL netlist from the synthesis tool will contain components that are specific to the target technology that was chosen during synthesis. Typically, the digital logic described in VHDL are meant to be implemented on PLDs from vendors such as Xilinx, Altera, Lattice *etc.*and hence they become the target technology during the synthesis process.

The gate level VHDL netlist generated can be optimized for performance and would then contain components specific to such PLD vendors. However, the default PSpice model library does not contain digital devices from these vendors and the gate level VHDL description offer tremendous challenge while translating into its equivalent PSpice circuit file. Considering these factors the digital logic described in VHDL is synthesized based on a target technology that is easier to translate into a PSpice circuit file and also contain digital devices that are either currently available or that can be modeled in PSpice A/D. The gate level netlist (thus produced) may not be optimized for performance but it is relatively easier to translate such gate level netlist into its equivalent PSpice circuit file.
Although the proposed design methodology uses Synplify as the synthesis tool, mixed level simulation doesn't limit to any particular synthesis tool and the same functionality could be achieved across various tools if suitable changes are made to the interfacing program that translates the gate level VHDL netlist into its equivalent PSpice circuit file.

3.1.1 Design methodology for mixed level simulation in PSpice A/D

In the previous section, a new design methodology was presented. A typical design flow for this methodology is as follows

1. Digital logic in VHDL (top down design methodology) is verified using logic simulators like ModelSim, NC VHDL. See section2.2.

2. The RTL VHDL code is synthesized in Synplify using Lattice MACH 111 as the target technology, see figure3.2. This results in a gate level VHDL netlist which contains devices specific to Lattice MACH 111 [13].

3. The gate level netlist is now converted into a PSpice circuit file using the interfacing software which is developed as a part of this research.

4. The Circuit file is converted into a OrCAD symbol which can be placed on a schematic editor along with analog components and mixed signal simulations can thus be performed.

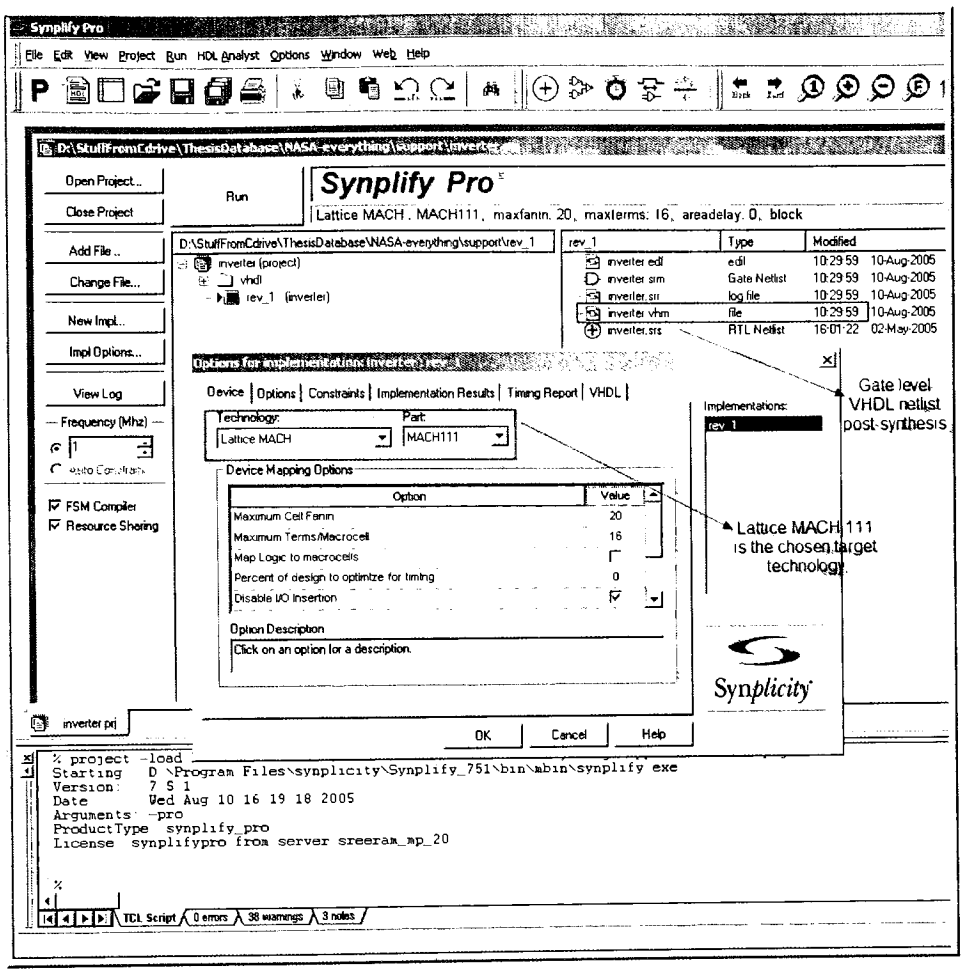


Figure 3.2: Synthesis using Syplify

This concept clearly enables parallel designing of analog and digital architectures, as well as total system verification in PSpice A/D. Total system verification reduces the time to debug any interface problems in the design and hence the time to market the final product is reduced. Sections 3.2.1 and 3.2.2 will elucidate the process involved in translation of the gate level VHDL netlist into a PSpice circuit file.

3.1.2 Lattice MACH 111

Lattice Semiconductor Inc, is one of the leading manufacturers of programmable logic devices and their MACH 111 family offer high-performance, low cost Complex Programmable Logic Devices (CPLDs). In the proposed methodology (see figure 3.1) the VHDL description of the digital logic, (which is at register transfer level) is targeted at LATTICE MACH 111 during synthesis. Section 3.2.1 summarizes the list of logic primitives of the LATTICE technology that could be modeled in PSpice A/D.

The MACH 111 consist of two Programmable Array Logic (PAL) blocks interconnected by a programmable switch matrix. The two PAL blocks are complete with product term arrays and programmable macro cells, which can be programmed as high speed or low power. The switch matrix connects the PAL blocks to each other and to all input pins, providing high degree of connectivity between the fully connected PAL blocks. This allows designs to be efficiently placed and routed. The MACH 111 macrocells can be configured as either registered or combinatorial, with programmable polarity. The flip flops can be configured as D Type or T Type, allowing for product term optimization. The flip flops can also be asynchronously initialized with common asynchronous reset and preset product terms. In this thesis, the flip flops are configured as D Type [12].

Synthesizing with Lattice MACH technology results in a gate level description, that describes the digital logic in terms of simple sum of product terms. This is relatively easier to comprehend, and it facilitates translation of gate level description into a PSpice circuit file.

3.2 Conversion of VHDL into PSpice circuit file

The translation of the gate level VHDL netlist into its equivalent PSpice circuit file requires

1. A library of PSpice models for Lattice MACH components.
2. An Interfacing software that utilizes components from this library and create a PSpice circuit file from the gate level VHDL netlist.

3.2.1 PSpice library of Lattice devices

The gate level VHDL netlist generated by the synplify (synthesis tool) contains components specific to Lattice MACH technology. The following set of combinational and sequential logic elements from the MACH 111 family are utilized by the interfacing software during the translation process. These Lattice devices are modeled in PSpice by customizing existing digital primitives. The customized models are created within a new PSpice library called ICreated.LIB

Combinational logic elements

1. IBUF Input Buffer
2. OBUF - Output Buffer
3. INV - Logic inverter
4. OR2 - 2 Input logic OR
5. XOR2 - 2 Input logic XOR
6. AND2 - 2 Input logic AND

Sequential logic elements

1. MACHDFF Reset predominant D flip flop with low preset and reset
2. DFFRH - Reset predominant D Flip flop with preset remaining HIGH all times
3. DFFSH - Reset predominant D Flip flop with reset remaining HIGH all times
4. DFF Reset predominant D Flip flop

Figure 3.3 presents a comparison between a simple logic gate (AND) from the Lattice MACH 111 family, described in VHDL and its equivalent SPICE model that is currently available in PSpice A/D. The interfacing software replaces every instance of "AND2" in the gate level VHDL netlist with its equivalent PSpice model in the PSpice circuit file that

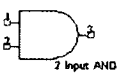
it creates. The customized PSpice model library (ICreated.LIB) is populated with such PSpice digital models that are functionally equivalent with its Lattice MACH counterparts in VHDL. For details about PSpice digital device modeling, please refer section ??.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.components.all;

entity AND2 is
port(
  O :  out std_logic;
  I0 :  in std_logic;
  I1 :  in std_logic);
end AND2;

architecture beh of AND2 is
  signal VCC : std_logic ;
  signal GND : std_logic ;
begin
  VCC <= '1';
  GND <= '0';
  O <= I0 and I1; --  after 100 ps;
end beh;
```

AND logic of a Lattice MACH device in VHDL

```
*
* 2 INPUT AND GATE
*
.SUBCKT AND2  I0 I1  O
+     optional: DPWR=$G_DPWR DGND=$G_DGND
*
U1 AND(2)
+  DPWR  DGND
+  I0 I1  O
+  D_PLD_GATE IO_PLD
*
.ENDS
```

PSpice model of the AND logic



Schematic representation of the PSpice AND logic

Figure 3.3: Lattice MACH AND logic in VHDL and its equivalent PSpice model

However, the PSpice default libraries do not contain equivalent SPICE models for the sequential logic elements from the Lattice MACH family. Before the interfacing program can replace the sequential logic elements in the gate level VHDL netlist, PSpice models for such sequential elements must be created. In this case, a new sequential element is created in PSpice by modifying some existing primitives to match the logic described in VHDL. Figure 3.4 shows the VHDL model of a sequential element - D Flip Flop, based on which other Lattice MACH devices are built. Hence, the PSpice equivalent of this element will become the basic building block for the remaining sequential elements in the customized PSpice model library.

Figure 3.5 displays the results of simulating the VHDL model of the D flip flop using Model Sim as the digital simulator.

This is a RESET dominant D Flip Flop. Table 3.1 is constructed based on this result.

Figure 3.6 represents the result of simulating a PSpice model of an existing D flip flop,

```
--Copyright (c) 1997 by Synplicity, Inc.  All rights reserved.
library ieee;
use ieee.std_logic_1164.all;
entity prim_dff is
    port (q : out std_logic;
          d : in std_logic;
          clk : in std_logic;
          r   in std_logic := '0';
          s : in std_logic := '0');
end prim_dff;

architecture beh of prim_dff is
begin
    ff : process (clk, r, s)
    begin
        if r = '1' then
            q <= '0';
        elsif s = '1' then
            q <= '1';
        elsif rising_edge(clk) then
            q <= d;
        end if;
    end process ff;
end beh;
```

Figure 3.4: The Primary D Flip Flop in VHDL

Clk	R	S	D	Q
Don't Care	High	Don't Care	Don't Care	Low
Don't Care	Low	High	Don't Care	High
Rising Edge	Low	Low	Low	Low
Rising Edge	Low	Low	High	High

Table 3.1: Truthtable of the D Flip flop

from one of the default libraries. Although this flip flop is not exactly similar to the VHDL model shown earlier, it can be made to function like its VHDL counterpart making suitable changes.

The current PSpice model returns an invalid condition when it receives a logic High on both Reset and Set. Table 3.2 represents the desired logic states in Reset (R') and Set (S') inputs of the PSpice D flip flop.

Figure 3.7 presents a schematic representation of the PSpice D flip flop with its Reset/Set inputs being treated appropriately to avoid the invalid condition. This modified D flip flop is now functionally equivalent to its VHDL counterpart. This is verified in PSpice and simulation results are presented in figure 3.9. Figure 3.8 represents the PSpice subcircuit

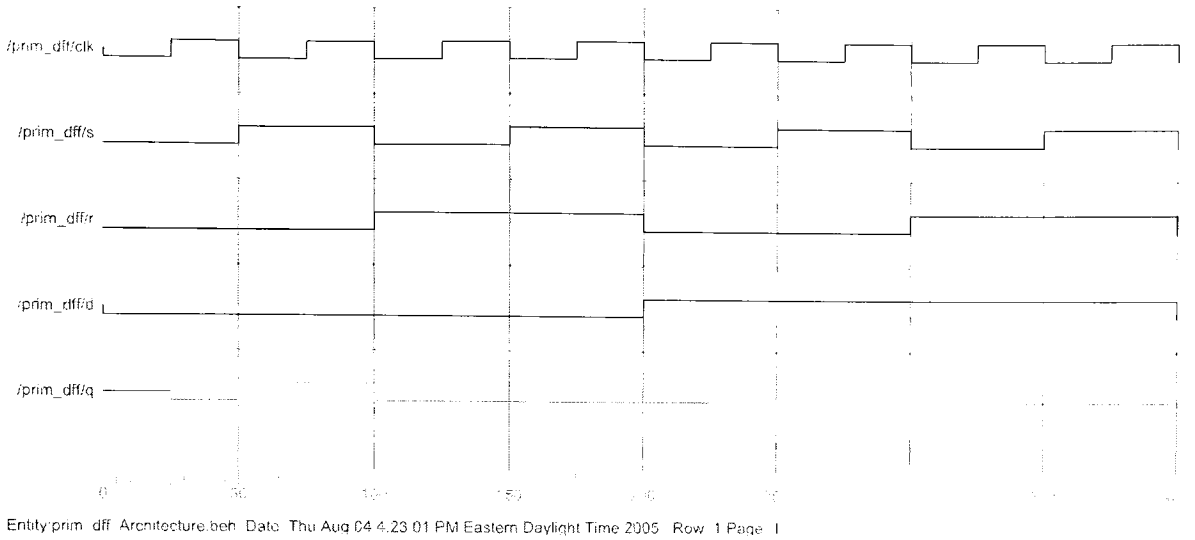


Figure 3.5: Simulation of Prim DFF in VHDL

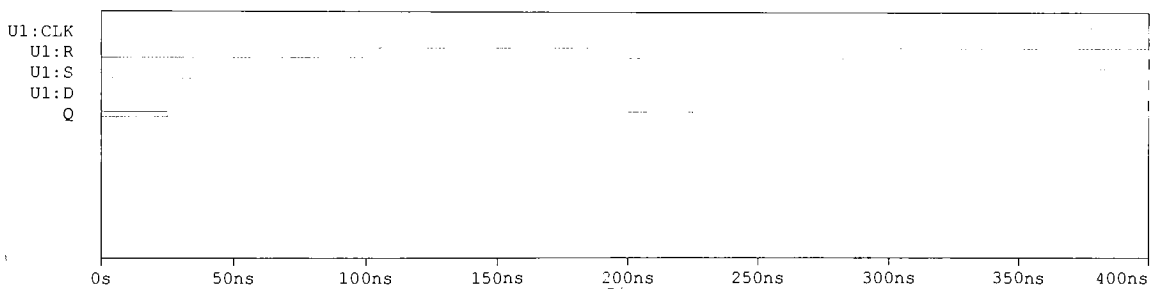


Figure 3.6: Simulation of an available PSpice model of a D Flip flop

notation of the modified D flip flop.

Similarly, other sequential components in the Lattice MACH family are simulated in VHDL, and then their functionality is modeled in PSpice. Once the required logical functionality is attained in PSpice, the digital device model is added to the custom library (Icreated.LIB). The interface program can then instantiate these custom built PSpice digital devices at appropriate times in the final PSpice circuit file that represents the gate level VHDL netlist.

R	S	R'	S'
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Table 3.2: Table for desired Reset and Set conditions in PSpice D flip flop

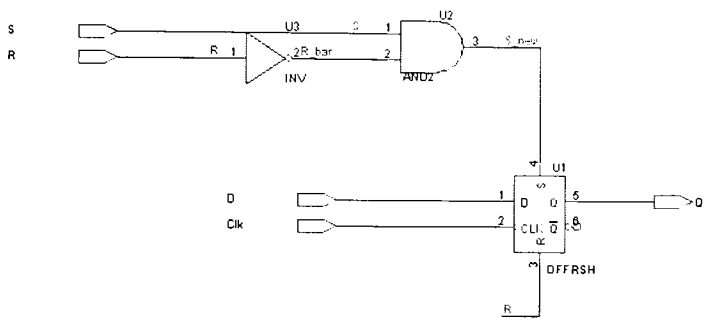


Figure 3.7: Schematic representation of Prim DFF in PSpice A/D

3.2.2 Interfacing software

The design methodology presented in section 3.1 utilizes an interfacing software to translate gate level VHDL description into a PSpice circuit file. Converting a gate level VHDL netlist to a PSpice circuit file is necessary to perform mixed level simulation in PSpice A/D. In section 3.2.1, the creation of PSpice models for Lattice MACH digital devices was discussed. The interfacing software reads the gate level VHDL netlist, identifies a Lattice MACH device and replaces it with its equivalent PSpice model. It then writes a new PSpice circuit file in which the entire gate level VHDL netlist is treated as one huge subcircuit file. Figure 3.10 presents a flow chart followed by the interfacing software to perform this translation.

The following procedure is followed by the interfacing software

1. Parse through the gate level VHDL netlist and skip until the Main Entity within the file is reached.

2. Within the Main Entity, extract the name of the inputs and outputs. If the inputs/outputs

```
.SUBCKT DFFRSH_PE Clk D Q R S
X_U1          D_CLK R S_NEW Q M_UNO001 $G_DPWR $G_DGND DFFRSH
X_U2          S R_BAR S_NEW  $G_DPWR $G_DGND AND2
X_U3          R R_BAR  $G_DPWR $G_DGND INV
.ENDS

.SUBCKT DFFRSH D CLK R S  Q QBAR
+     optional: DPWR=$G_DPWR DGND=$G_DGND
*
U1 INVA(2)
+  DPWR  DGND
+  R    S
+  RBAR SBAR
+  D_PLD_GATE IO_PLD
*
U2 DFF(1)
+  DPWR DGND
+  SBAR RBAR  CLK
+  D
+  Q
+  QBAR
+  D_PLD_EFF IO_PLD
*
.ENDS
```

Figure 3.8: Reset dominant D flip flop in PSpice that is equivalent to the VHDL model

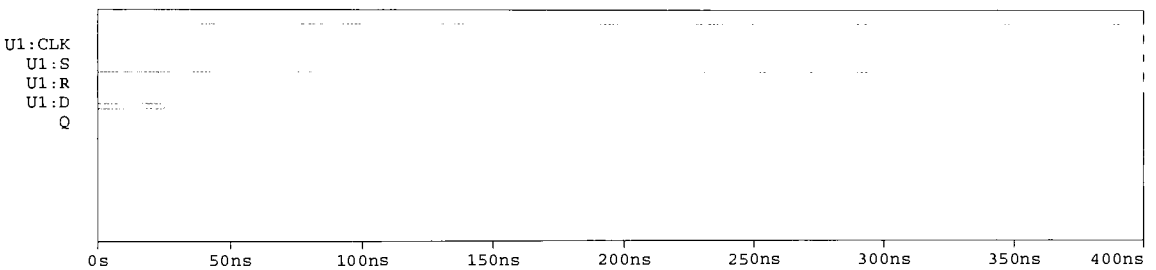


Figure 3.9: Simulation of PSpice D Flip flop after its Reset/Set inputs are gated

are declared as a BUS, elaborate the bus entries and assign individual net names for each one of the bus entries. PSpice digital device modeling language does not permit BUS declaration.

3. Using the input and output names obtained, define the subcircuit header in the PSpice circuit file by following the PSpice modeling language syntax. See section **??**.

4. Continue to parse the VHDL file and skip until the Main Architecture of the entity is reached.

5. Within the Main Architecture, skip the section where internal signal and component names are declared.

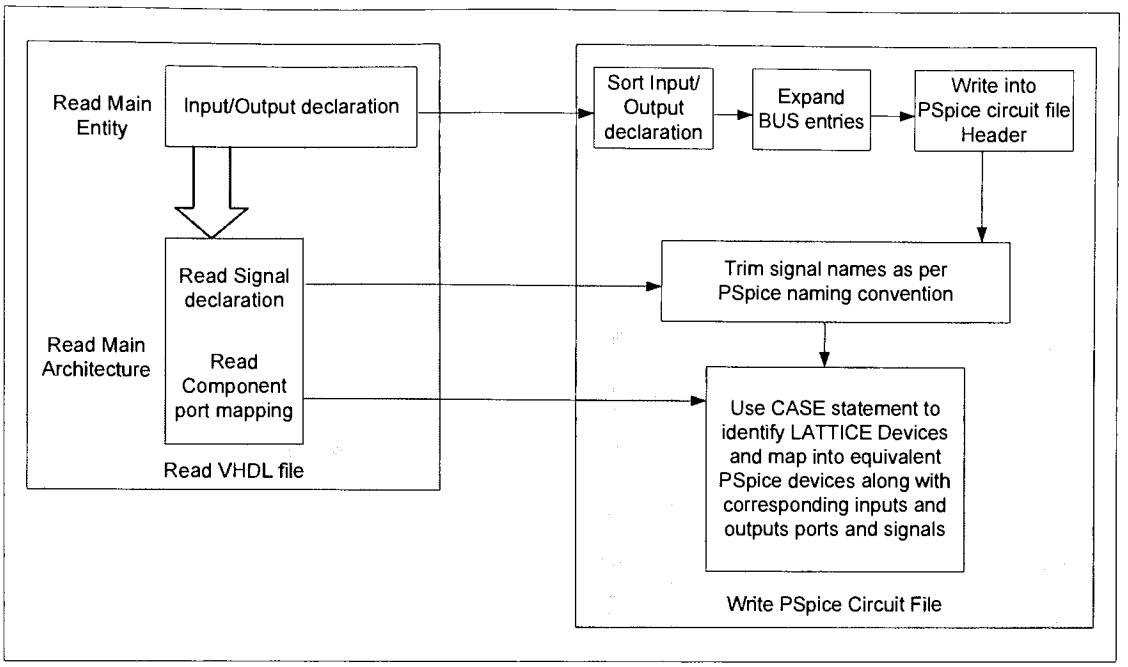


Figure 3.10: Flow chart for VHDL-PSpice Conversion program

6. Scan the architectural definition and identify the Lattice MACH device that is being "port - mapped"

7. Use "CASE" statements to map the identified component with its equivalent PSpice model.

8. Scan the "port - mapping" definition to identify the input and output net names and assign them to appropriate PSpice model terminals.

9. Loop until the end of architecture section is reached.

Figure 3.11 shows the user interface of the Interfacing program.

3.2.3 Simulation setup in PSpice A/D

In sections 3.2.1 and 3.2.2, discussions on PSpice library of Lattice MACH devices and PSpice circuit file were presented. The PSpice circuit file provides a textual description

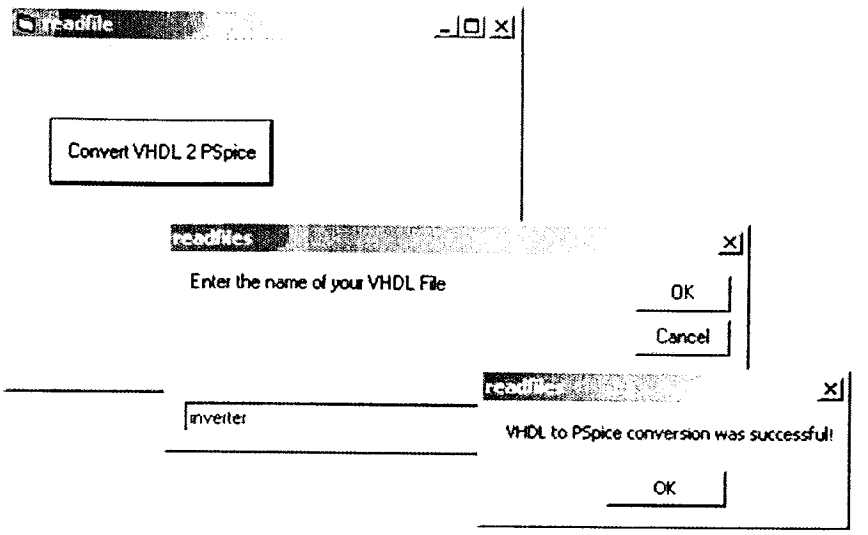


Figure 3.11: The user interface of the Interfacing software

of a circuit. A circuit file cannot be connected to other components that are present on a design schematic. A circuit design in PSpice is created by inter-connecting schematic symbols representing electrical components. Therefore it is necessary to create a schematic symbol that represents the circuit file.

Creating schematic symbol will enable the circuit designer to place the component on a schematic and provide proper connectivity with other devices that constitutes the design. Using the Model import wizard, a PSpice accessory (see figure 3.12), a schematic symbol representing the textual circuit file is created. It is also required to re-configure PSpice simulation parameters such that the simulator recognizes the device models of Lattice MACH devices that were designed as a part of this thesis.

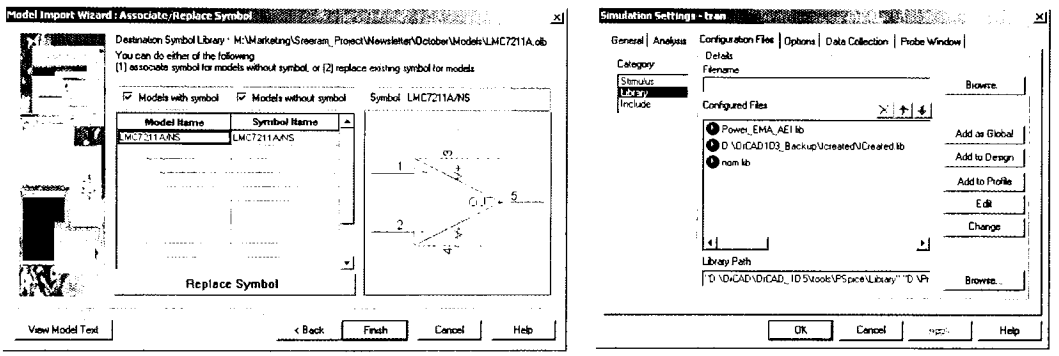


Figure 3.12: Model import wizard and simulation settings window in PSpice A/D

Chapter 4

Case study: A Mixed Signal design from NASA Goddard Space Flight Center

4.1 Overview of the design

Motion control and positioning of many space mechanisms are accomplished by using brushless motors. They are simple yet robust, offer high power to weight ratio, low inertia and optimal performances at high and low speed. Typical applications are scan mirrors, thrust vector control actuators, fuel value control actuators, solar array deployment, control moment gyroscopes, applications requiring light weight, low thermal emission, high and low rotation per minute. [6].

An overview of a mixed signal design involving an Actel Flash FPGA for controlling the motion of a brushless motor that runs a gas compressor for a space application at NASA Goddard Space Flight Center is shown in figure 4.1. This solution provides a fully synchronous design with single clock domain, higher flexibility due to the possibility of modifying or updating the system, cost reduction and better reliability, lower chance of ground bounce or system noise due to reduction in the simultaneously switching outputs.

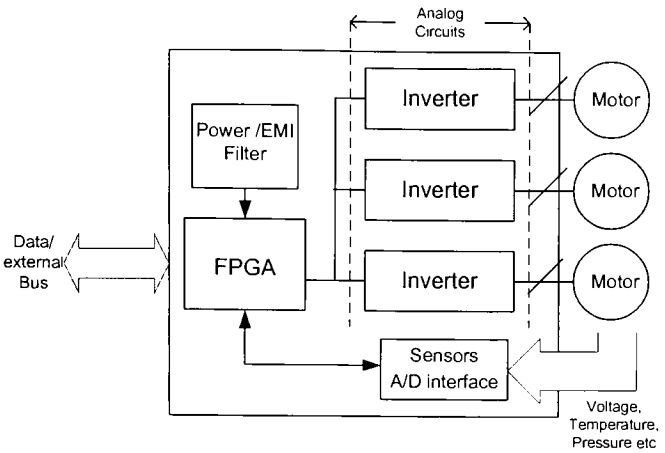


Figure 4.1: Block diagram of the design from NASA

4.1.1 Problem description

The motor is driven by digital circuits (modeled in VHDL) within a FPGA which controls and communicates with the main spacecraft computer. The analog section is an inverter with half bridge drivers and transformers, that provide the required current conditioning and 120 degrees out of phase waveforms with required power levels to run the motor.

The research team in NASA currently employ two different software simulators for this design. The digital logic within the FPGA is programmed in VHDL and simulated using an HDL simulator. The analog circuits in the inverter section are simulated in PSpice A/D. The inverter circuit modulates the duty cycle of the motor depending upon the input frequency. The input frequency is voltage dependent and the FPGA controls the voltage-to-frequency variation.

The verification of the complete design that consists of digital logic modeled in VHDL, and analog circuits designed using PSpice A/D is necessary to predict the overall system behavior. The differences in design methodologies and EDA tools utilized during the design process makes total system verification very difficult. The ability to simulate VHDL models in PSpice is therefore required to solve the predicaments in total system verification, as it combines two different design procedures (*traditional and top down*) in one simulation package (*PSpice A/D*).

4.1.2 Simulation of the Behavioral VHDL model

The VHDL model that is used in this design, not only to controls the speed of the motor but also to co-ordinate other communication protocols between the FPGA and the space-craft's main computer. The waveforms shown in the figure 4.2 is obtained after simulating the VHDL code for 3 milliseconds. Although the waveforms indicates all the input and output parameters in the VHDL program, the inputs and outputs that are of concern for this example are listed in table 4.1.

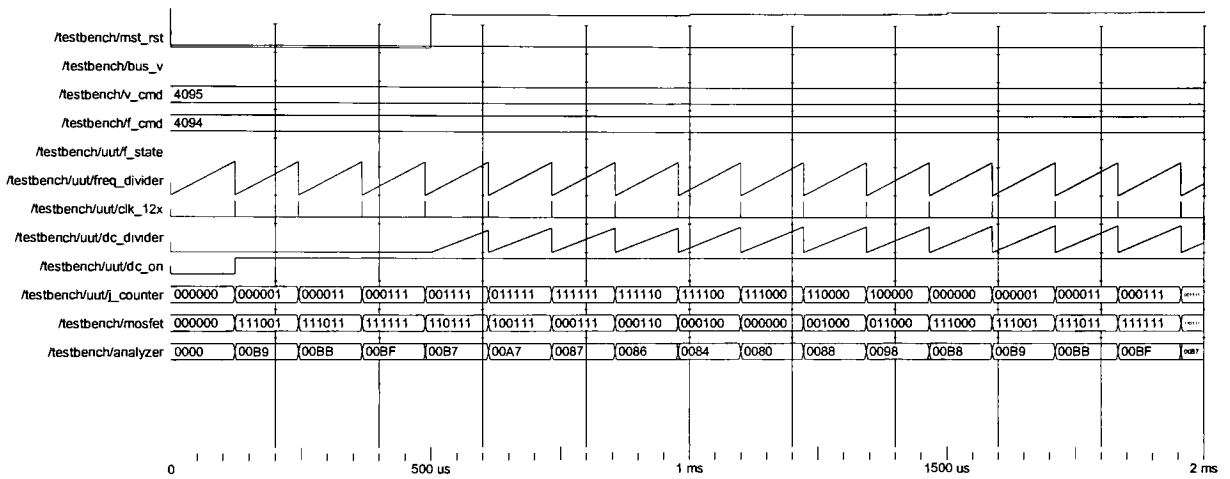


Figure 4.2: Behavioral simulation of the VHDL model

The purpose of this VHDL program is to advance a Johnson counter according to a set of design rules and specifications. The functionality of this program can be understood as follows

1. When Master reset = 1, initialize all inputs and outputs
2. When reset = 0, rising edge of the clock and Clk_12x = 1
    - Increment the 6 bit Johnson counter
    - dc_on = 1
    - dc_divider = 0

Name of Input/Output	Definition
main_clk	Main clock
mst_rst	Master reset
clk_12x	When true, advance Johnson counter
v_cmd	Voltage command
bus_v	Input bus voltage
freq_divider	Counter for frequency
f_ramp	Frequency ramp control
f_cmd	Frequency command
f_state	Internal ramped frequency
mosfet	6 bit output to inverter/analog circuitry
dc_divider	Duty cycle divider
dc_on	If true, duty cycle is enabled

Table 4.1: List of relevant input and outputs from the VHDL model

3. When reset = 0 and on rising edge of the clock

   - If f_ramp = 1,

     freq_divider = freq_divider + f_cmd until freq_divider is less than 10000000, after which Clk_12x = 1

   - Increment dc_divider with bus voltage until dc_divider is less than the product of v_cmd times 4096 after which turn duty cycle "OFF"

   - Ramp f_state one count at a time until it equals f_cmd

4. "mosfet" which is the final output, is the output of Johnson counter gated along with duty cycle control.

The output at "mosfet" is the input to the analog section, hence it is important to co simulate the VHDL model along with the analog circuits to enhance identification of any interface issues.

4.2 Mixed level and mixed signal simulation

The following sections demonstrates the proposed design methodology.

4.2.1 Step 1: Gate level simulation in VHDL

The behavioral VHDL model is synthesized in LATTICE MACH 111 technology using Synplify (synthesis tool). The gate level VHDL netlist (description) contains components from the chosen technology. This netlist is validated by subjecting it to simulation. Figure 4.3 presents the result after simulating the gate level VHDL description. This result is compared with the results from behavioral simulation (section 4.1.2) before proceeding further. Figure 4.4 presents the RTL view of the VHDL model (obtained from Synplify during logic synthesis).

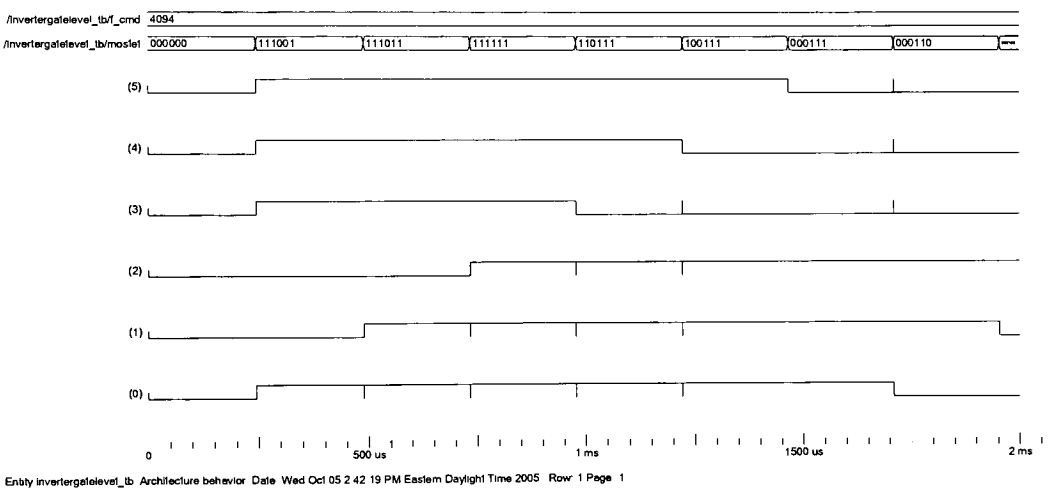


Figure 4.3: Gate level simulation of the VHDL model in ModelSIM

Table 4.2 provides a list of components from the Lattice MACH family of devices, present in the gate level VHDL description. PSpice models for these Lattice MACH devices were created as a part of this thesis (section 3.2.1). The gate level VHDL netlist is now ready for translation into PSpice circuit file.

4.2.2 Step 2: Simulation of VHDL model in PSpice A/D

Using the interfacing tool (discussed in section 3.2.2), the gatelevel VHDL model is converted to a PSpice circuit file. PSpice model editor is then used to create a circuit symbol

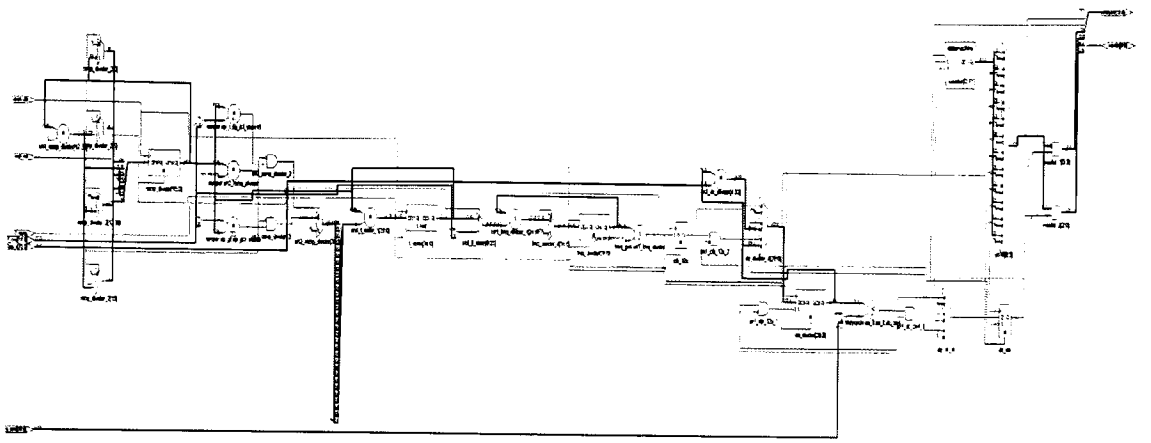


Figure 4.4: RTL view of the VHDL model after synthesis

Gate	Instances
DFFRH	121
DFFSH	2
IBUF	39
OBUF	22
AND2	879
XOR2	181
INV	678
OR2	52

Table 4.2: Gate primitives

which then can be placed on the schematic along with other analog or digital components. Figure 4.5 shows the circuit representation of the VHDL model after synthesis. The PSpice model representing the gate level VHDL netlist is subjected to simulation, similar to its behavioral counterpart by connecting the inputs to appropriate voltage levels (by referring to the testbench code). Figure 4.6 shows the results of simulation in PSpice and the waveforms at the 'mosfet" pin and it matches the post synthesis results observed in section 4.2.2

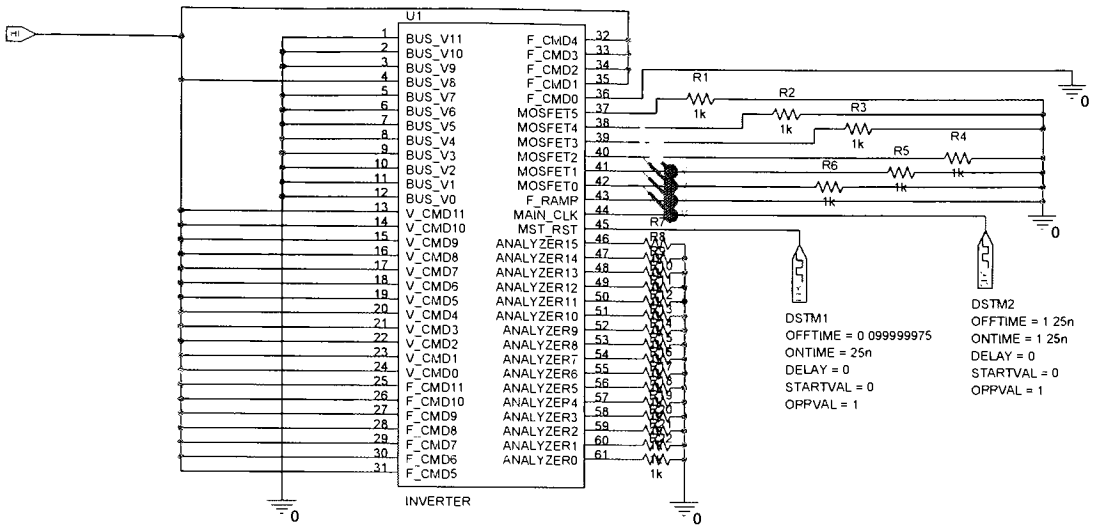


Figure 4.5: PSpice circuit representation of the VHDL model

4.2.3 Step 3: Integration of digital signals from the VHDL model with the analog circuit

Currently, the VHDL model is simulated on a different tool (Model SIM) than the analog circuit, which is designed and simulated in PSpice A/D. Since HDL simulators (Model SIM) and PSpice A/D are not integrated together, co-simulation of the analog and digital sections of the design is not possible. This forces the designers to make certain assumptions in their simulation methodology which is not the ideal or recommended way.

By following the proposed methodology, the entire VHDL model is converted into a PSpice circuit element which can be connected to the analog circuits. By doing so, cosimulation of the analog and digital components is now possible in one software tool, which is PSpice A/D. This would increase the accuracy of results obtained after simulation, because the co simulation guarantees precise arrival times of input signals from the VHDL model. The outputs from the PSpice equivalent VHDL model is given as the input to the analog circuits. Section of this circuit is shown in figure 4.7. Figure 4.8 shows the result of the mixed signal simulation with the PSpice equivalent of the VHDL model along with the

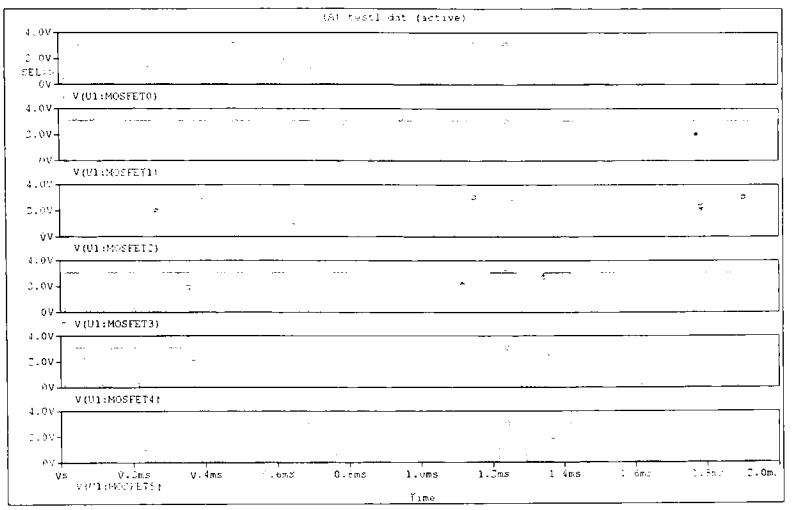


Figure 4.6: Results after PSpice simulation of the VHDL (gatelevel) model

analog circuits. This result represents the output of the inverter which converts DC power to AC after receiving appropriate control signals from the VHDL model. These results were acceptable by the engineers in NASA Goddard Space Flight Center.

4.3 Summary

Gate level simulation of the VHDL model (section 4.2.1) and its PSpice equivalent circuit simulation (section 4.2.2) produced exactly the same results. This indicates

1. Accurate conversion from VHDL to PSpice circuit file
2. Customized PSpice library of devices for the LATTICE technology is modeled accurately

Using the PSpice circuit equivalent of the VHDL model in the mixed signal design reduces speculation of the arrival times of the input signals and hence reduces possibility of mismatch, enhances the ability to identify interfacing problems and enables entire system verification in one software medium.

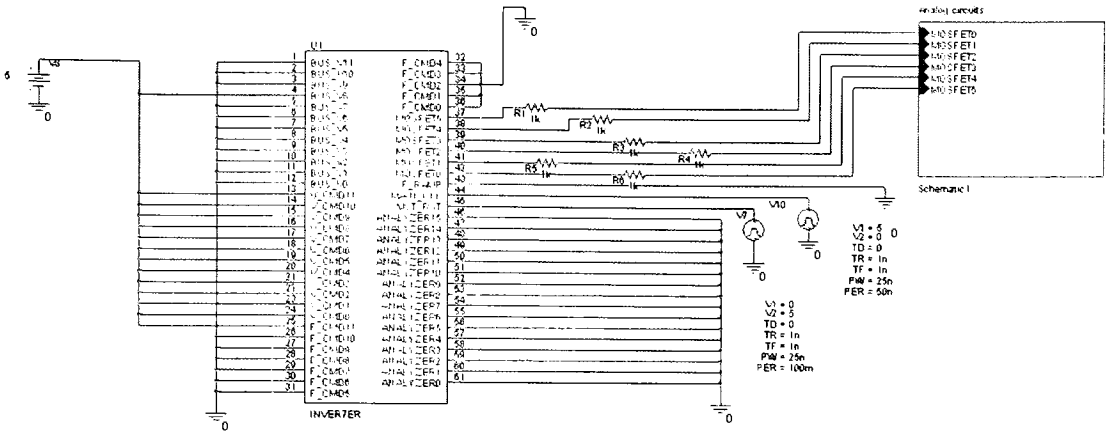


Figure 4.7: Circuit representation of VHDL code during Co simulation

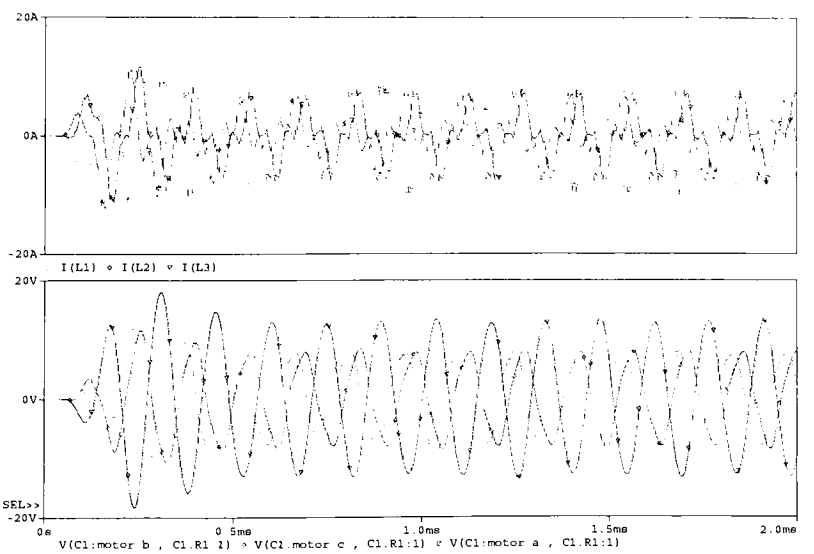


Figure 4.8: Three phase current and voltage waveforms after mixed signal simulation with VHDL model in PSpice A/D

Chapter 5

Conclusions and Future Work

In this work the possibility of mixed level simulation in PSpice A/D using VHDL was investigated. To achieve this goal the following techniques were followed

1. Generating gate level VHDL netlist from RTL VHDL using Synplify (logic synthesis tool).
2. Creating a PSpice subcircuit definition that represents the gate level VHDL description (*Interfacing program*).
3. A PSpice library of was developed to assist the mixed signal simulation by providing necessary digital models (for Lattice MACH 111 devices)

A mixed signal design from NASA Goddard Space flight center was used to validate the accuracy of the developed techniques. The results of using the proposed methodology in this design example can be summarized as follows

1. Simulation of the entire mixed signal system which includes the FPGA representation as well as the analog circuits.
2. The arrival times of the signals from the FPGA were modeled accurately which produced results which were realistic than before.

Overall, the proposed methodology shows an effective way to verify the functionality of the mixed signal design. In examples such as the one studied in this work, different design

teams work on different sections of the same design, namely analog and digital. In such situations, where the possibility of error introduced in the design process is high, the proposed methodology provides an easy and simple way to verify the functionality of the entire mixed signal design and facilitates the identification of integration problems earlier in the design cycle. Other areas where the proposed design methodology could be applicable are, design of electro mechanical systems, power systems design, signal processing applications *etc.*and other design areas where programmable logic devices are actively becoming a part of PCB designs.

5.1 Future Work

The following problems can be identified as suitable for further research

1. Increased simulation time. Using gate level VHDL netlist becomes a bottleneck if the number of gates in the netlist exceeds a few thousand. In order to reduce the simulation time, further investigation on behavioral simulation methods of the VHDL code in PSpice is necessary.

2. MATLAB SIMULINK is a system level simulation tool. It has the ability to integrate with PSpice simulation engine as well as with digital simulation tool such as MODEL SIM. If all these three tools can be integrated seamlessly, they can provide co simulation environment between system level, circuit level and HDL based designs.

3. Automatization and graphical user interface. The proposed methodology involves repeating a sequence of commands in each of the EDA tools, which can be scripted and automized. In doing so, a lot of details can be hidden from an end user which would increase the appeal of the solution. A graphical user interface which would take in the VHDL codes along with its test bench inputs all the way to a PSpice circuit symbol will be the utlimate comfort in this solution.

Bibliography

[1] Peter Ashdon. *The Designers Guide to VHDL 2nd Edition.* Morgan Kaufmann, United States of America, 2002.

[2] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for Deep Submicron FPGAs.* Kluwer Academic, United States of America, 1999.

[3] Dave Brady and Tom Dewey. Are the benefits of using FPGA's consumed by the obstacles of integrating the FPGAs on Printed Circuit Board? *Mentor Graphics White Paper*, 2003.

[4] Cadence. *PSpice Reference Guide*, 1998. Version 10.5.

[5] Cadence. *PSpice User's Guide*, 2005. Version 10.5.

[6] G.C. Caprini, F. Innocenti, L. Fanucci, S. Ricci, G. Taraschi, P. Terreni, M. Tonarelli, and L. Tosi. Embedded system for brushless motor control in space application. *MAPLD International Conference*, pages 151 – 156, 2004.

[7] Nancy L. Eastman. Considerations for Mixed analog/digital PCB design. *WESCON/96*, pages 297 – 301), YEAR = 1996.

[8] B Fawcett. Synthesis for FPGAs: an overview. *WESCON/94. 'Idea/Microelectronics'. Conference Record*, pages 576 – 580, 1994.

[9] Peter Frey and Radharamanan Radhakrishnan. Parallel mixed-technology simulation. *Fourteenth Workshop on Parallel and Distributed Simulation*, pages 7 – 14, 2000.

[10] Bashir Al Hashimi. *The art of Simulation using Pspice Analog and Digital.* CRC Press, United States of America, 1995.

[11] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig. Design of mixed - signal systems - on- chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19:1561 – 1571, 2000.

[12] Lattice Semiconductor Corporation. *Lattice MACH 111 Reference Guide.*

[13] Synplicity. *Synplify Pro Reference Guide*, 2004. Version 7.5.1.